Preliminary Design Document

FSA Portal, Release 1

8/5/2004

Author:	Christopher_J_Lawson
Last Modified By:	Christopher_J_Lawson
Creation Date:	12/18/2001 9:15 AM
Last Updated:	1/31/2002 10:14 AM
Version:	1.1

Document Control

Change Record

Date	Author	Version	Change Reference
12/18/01	Chris Lawson	1.0	Beginning of document creation.
01/02/02	Chris Lawson	1.0	Modified after review w/ Matt Wilson.
01/25/02	Chris Lawson	1.0	Removed and changed document based upon new requirements.
1/28/02	Chris Lawson	1.1	Removed references to Release 2.
1/29/02	Chris Lawson	1.1	Removed component factory usage
1/31/02	Chris Lawson	1.1	Added new process flows

1 Introduction

1.1 Purpose

A portal is an aggregation point for content, functions, and features using web-based technology with a unifying theme. Portals need to be able to display both unstructured and structured data. Unstructured data consists of documents found in intranet sites, Internet sites, document management systems, groupware databases, and network file systems. Structured data consists of data found in data warehouses, Enterprise Resource Planning (ERP) systems, legacy business data systems, and other databases.

A portal is a thin architecture layer integrating many different types of applications and services as shown in Figure 2. As portal tools evolve, additional functions previously provided by separate applications can be added to its capabilities (e.g., search, personalization, collaboration, etc.). Additionally, existing tools or software packages are beginning to include more functionality usually associated with portal tools.

The FSA Students and Financial Partners channels portals will bring together, in one simple, personalized Web page all the information and productivity tools relevant to FSA's customers, employees, and partners to make informed financial aid decisions and empower financial partners to assist students. The personalized "front door" will automatically identify and distribute content relevant to each user. The portals will integrate with existing FSA web sites (e.g., FAFSA, NSLDS, DLSS, etc..), and external sites (ELM Net, Meteor, etc..), using the ITA infrastructure. The portals will be the glue that bonds all of FSA's web services together providing a uniform starting point for students and financial partners to access FSA.

The purpose of this document is to provide a high-level overview of the features, design and functionality of the FSA Portal's Students Channel. Some of the information covered in this document is common between both the Students and Financial Partners Channels. Thus, a reference may be made to both Channels in certain sections of the document.

1.2 Scope

The scope of this document will present FSA with the framework for building a unified portal for students, parents, and financial partners to access FSA Financial Aid information. Release 1 includes the design of the enterprise portal framework with channel specific views for students and financial partners, the gathering of detailed requirements for both the students and financial partners channel specific views. Release 1 will also include the development of the FSA enterprise portal with channel specific views for both Students and Financial Partners to be deployed upon successful testing and approval. It will also be flexible to address the changing business environment needs of FSA. The end result of this project will be a long-term business architecture strategy for the FSA Enterprise Portal.

The Portals release strategy will include:

- **Release 1 (4 months after IRB approval)**
  - Build an enterprise portal infrastructure to enable the integration of re-engineered modernization systems
  - Build Students and Financial Partners views
  - Build link to Forms 2000 FP web site
  - Build link to FP Data Mart
  - Build links to Trade Associations
  - Build link to (FSA) University for FP conference registration
  - Provide regulatory content information for FPs
  - Analysis and definition of personalized data (both structured and unstructured) that will need to be accessible though the portal for financial partners
  - Re-use of common portlet services
  - Build search capabilities, internally and externally for FP views
  - Build content management for FP views
  - Ability to download Security Access Form for Data Mart and FMS
  - Create common uniform look and feel across the enterprise portal and sub-views
  - Perform usability testing to meet 508 compliance requirements
  - Develop on ITA infrastructure
  - Determine VDC hardware / software operation costs
  - Determine maintenance costs
  - Define standards for integration with Portal
  - Define on-going hosting strategy

1.3 Basic Business Needs

FSA's Internet channel has more than 35 web sites connected to multiple back-end systems. The FSA websites do not provide for a unifying theme or a consistent common look and feel across all sites. Students and Financial Partners do not have one single entrance point to access FSA's Internet services; they must access multiple URLs to retrieve financial aid information. FSA web sites need a personalized starting point for Students and Financial Partners to enter through one "front door" to access a single view of internally and externally stored content/information, application/services, business processes, and knowledge assets for every channel.

Business Problem:

- No single starting point for FSA customers
- No single view of information that can be personalized for Students or Financial Partners
- No integration across multiple websites and systems for internal and external use
- No uniform common look and feel for FSA web site(s)
- No consistent standards and architecture
- No common customer care component across all sites

1.4 Application Features & Benefits

Benefit Areas	Feature Description	Notes	Release #
Single Point of Entry	The portal will allow all end users to have a single entry point to information, resources, and tools they will need to perform their job.		1
Content Aggregation	The portal will aggregate information and resources to a central location.		1
Calendar	Calendars will be available to users. They will show current events, deadlines, training dates, and other information. This will provide these individuals, who enter through the portal, with knowledge of current tasks and deadlines. This will be updated by a member of FSA to ensure data is current.		1
Presentation	Provide a common look and feel for all views within the enterprise portal.		1
Survey	Allow users to answer specific questions that will generate feedback on what could be improved.		1

Search	Users can search across the portal and a number of other Ed sites to find related information.		1
Multi-Language	The portal will support both English and Spanish versions.	Users can toggle the language by interacting with the application.	1
Content Management	Content for certain areas of the portal will be centrally added. This will simplify content creation and distribution.	The first release will contain simple uses of this technology. Future releases could use advanced features in this area.	1
Interest Rate Information	Users will be able to easily find interest rate information.	The first release will be a simple representation of current interest rates.	1
Community Searching	Users can search for lenders, GA's, servicers, and Trade Associations.	Searching will be done alphabetically, by categories, or a key word and acronym search.	1
Contact Locating	Users can locate contact by selecting regions and/or states.		1

2 Application Architecture

2.1 Struts

Struts uses an adaptation of a common design pattern known as Model View Controller or MVC. (Struts actually uses MVC 2.) The system is broken down into three pieces, how the information is displayed (the view), how the information is maintained (the model), and rules and logic that define how the information is manipulated (the controller). By separating these pieces, it become easier to update any one piece, like choose to display the data in a different way, without affecting the others. The diagram below illustrates the components within Struts.

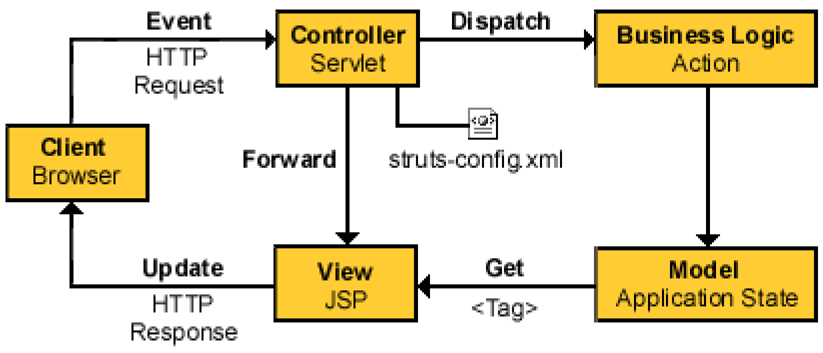

Figure 1

Struts overview

- **Client browser**
  This is the web browser that is used to reach the Portal. An HTTP request from the client browser creates an event. The Web container will respond with an HTTP response.

- **Controller**
  The Controller receives the request from the browser, and makes the decision

where to send the request. With Struts, the Controller is a command design pattern implemented as a servlet. The `struts-config.xml` file configures the Controller.

- **Business logic**
  The business logic updates the state of the model and helps control the flow of the application. With Struts this is done with an `Action` class as a thin wrapper to the actual business logic.

- **Model state**
  The model represents the state of the application. The business objects update the application state. ActionForm bean represents the Model state at a session or request level, and not at a persistent level. The JSP file reads information from the ActionForm bean using JSP tags.

- **View**
  The view is simply a JSP file. There is no flow logic, no business logic, and no model information -- just tags. Tags are one of the things that make Struts unique compared to other frameworks like Velocity.

Note: "Think thin" when extending the `Action` class. The `Action` class should control the flow and not the logic of the application. By placing the business logic in a separate package or EJB, we allow flexibility and reuse.

2.2 Blueprint

The application blueprint is designed to depict the application and all of its integration points into one view. It is not meant to be a detailed document; rather it shows a logical representation of the application and its pieces. The illustration below depicts the FSA Portal application, the different channels (or commonly called views), and its relative interactions. It can be seen that the Portal consists of three different channels. They are Schools, Financial Partners, and Students. All three will leverage the same base architecture and services but will use them in different capacities. The following will describe the diagram and the many layers and their interactions.

The top of the diagram depicts the portal layer and the different channels. These three channels target different audiences and thus offer different capabilities and functionalities. This will be evident in the presentation layer. The presentation, or user interface, will leverage Apache's Struts architecture. Struts, seen as the supporting layer for the UI, uses custom tag libraries and configuration files to help separate UI design from the rest of the application logic. Struts also helps glue the front-end to the logic and controls application flow and session management. Refer to the Struts section for a more detailed explanation of this architecture.

Below the UI reside the portals portlets. A portlet is a self-contained, functional "view" of information. It can be easily reused and really are the building blocks for the application's functionality. They are more tangible than the RCS pieces since they represent visual and tangible functional areas. Whereas RCS provides services such as database persistence which is invisible to the end user. Portlets offer functionality such as registration, searching, headlines, login, calendar, and feedback. They interact with services in the RCS suite to handle common application functionality (e.g. error logging and the like). They also interact with the presentation layer to provide visual information and data.

The Reusable Common Services provide common application services that many applications can leverage. They are designed to be highly reusable and in the diagram below they are shown to be connected with the portlets. This is to represent that these services (RCS) provide standard functionalities to the portlets. RCS is supported and was developed by ITA. Further information can be found in the Technical Architecture section of this document.

An Enterprise Application Integration layer is essential to provide data to a wide range of applications. Both the portlet layer and the RCS layer will leverage the EAI bus. However, it is important to note that the FSA Portal will NOT be leveraging the EAI bus to retrieve any information. Release 2 is planned to provide this data integration.

Lastly, the data layer depicts department-wide data sources. The portlets, RCS, and EAI will utilize databases which encompass large amounts of system data.

Please note that the Schools channel, in production, will be using the same general application architecture. However, the portlets that Schools is using are not current and differ functionally from those that will be incorporated in the Students and Financial Partners views.

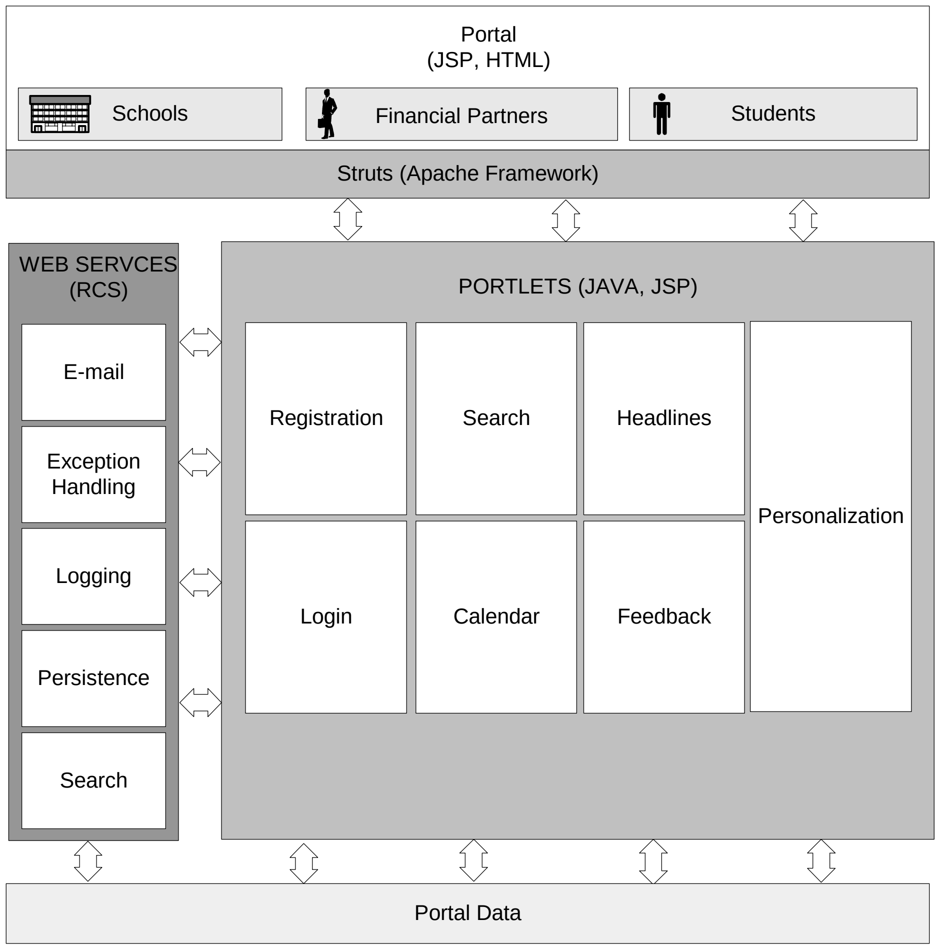

Figure 2

2.3 Page Navigation/Flow

NOTE: The following process flows are DRAFTS and will be updated this week.

2.3.1 High-Level Functional Map

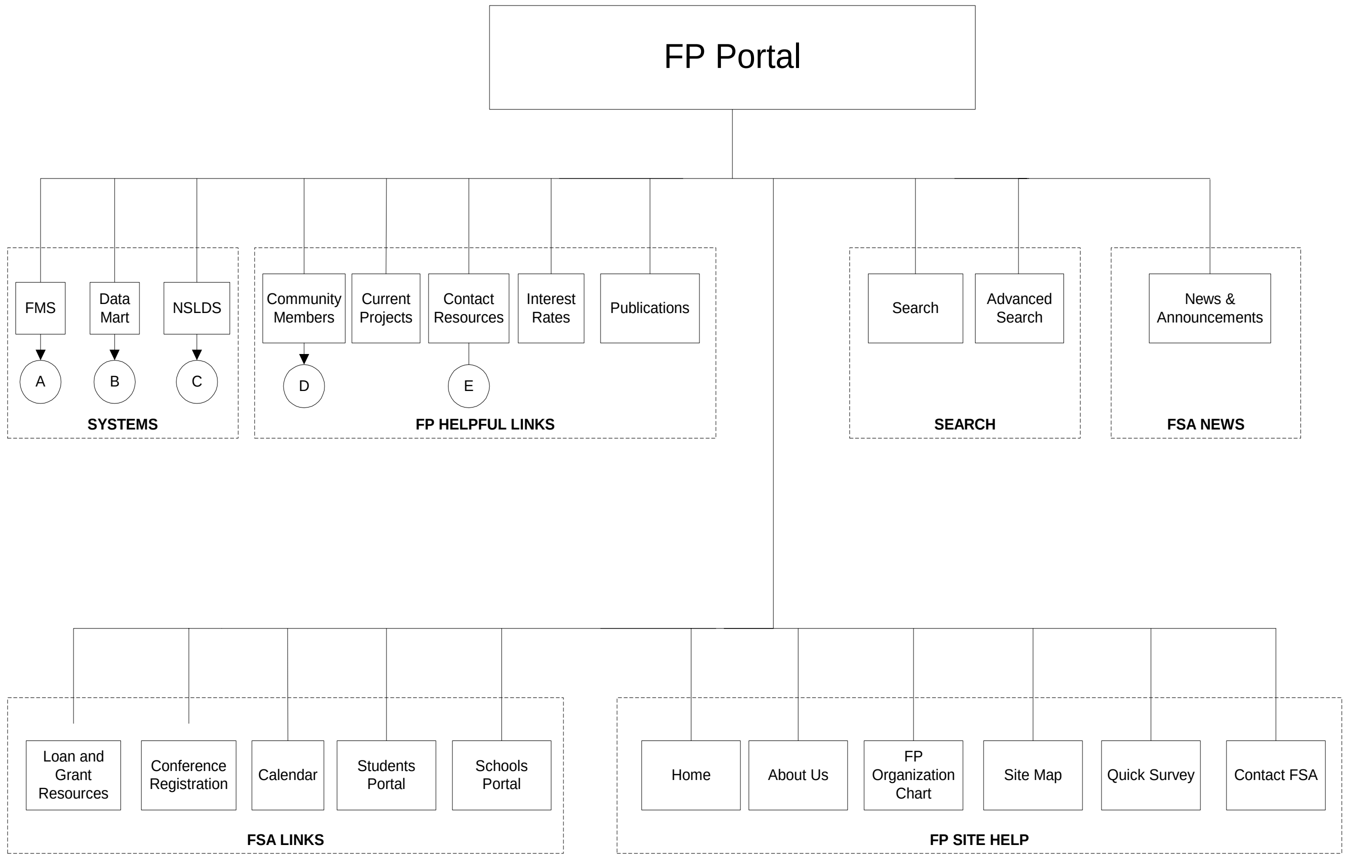

Figure 3

2.3.2 FMS Process Flow

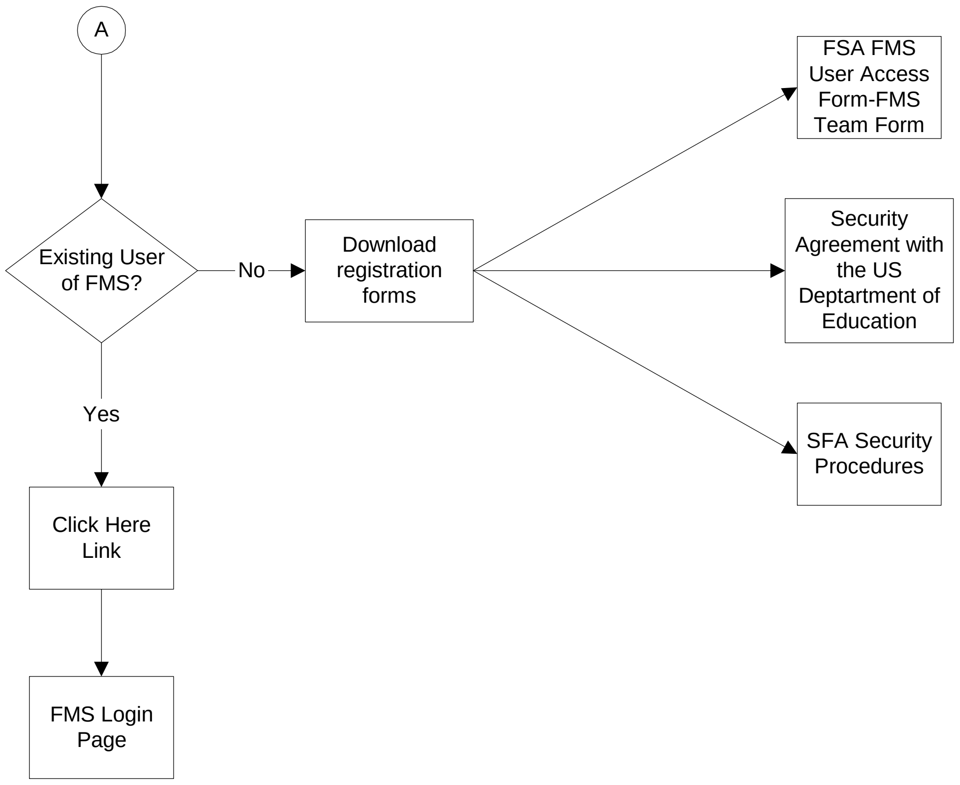
A
Existing User of FMS?
No
Yes
Download registration forms
FSA FMS User Access Form-FMS Team Form
Security Agreement with the US Deptartment of Education
SFA Security Procedures
Click Here Link
FMS Login Page

Figure 4

2.3.3 Data Mart Process Flow

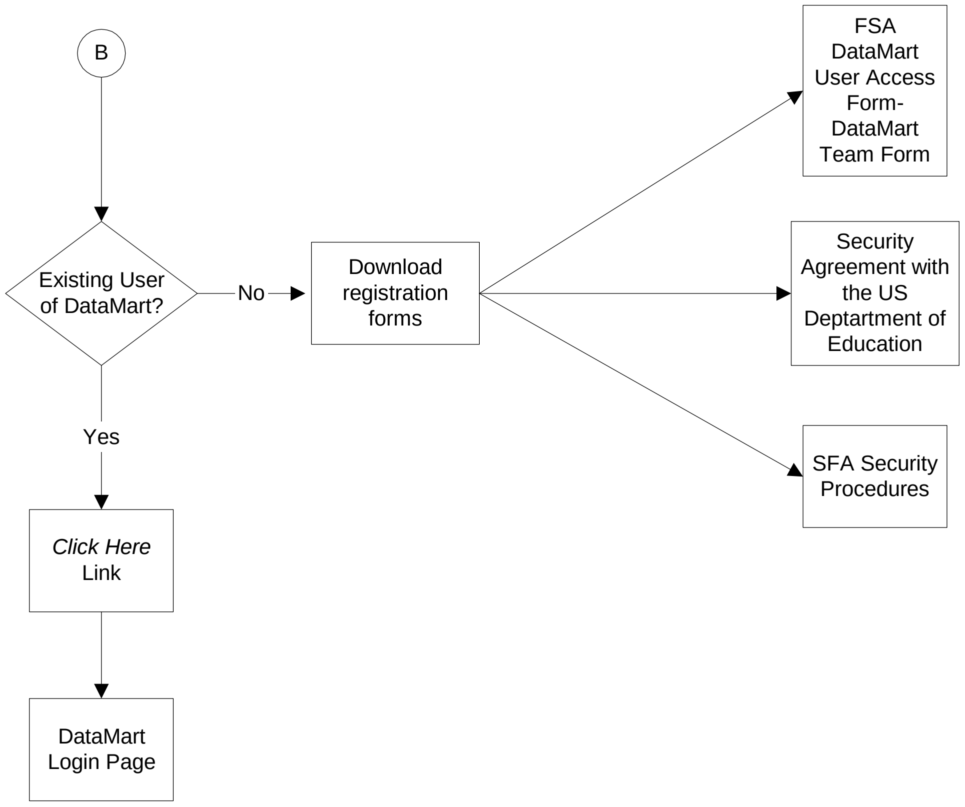
B
Existing User of DataMart?
No
Yes
Download registration forms
FSA DataMart User Access Form- DataMart Team Form
Security Agreement with the US Deptartment of Education
SFA Security Procedures
Click Here Link
DataMart Login Page

Figure 5

2.3.4 NSLDS Process Flow

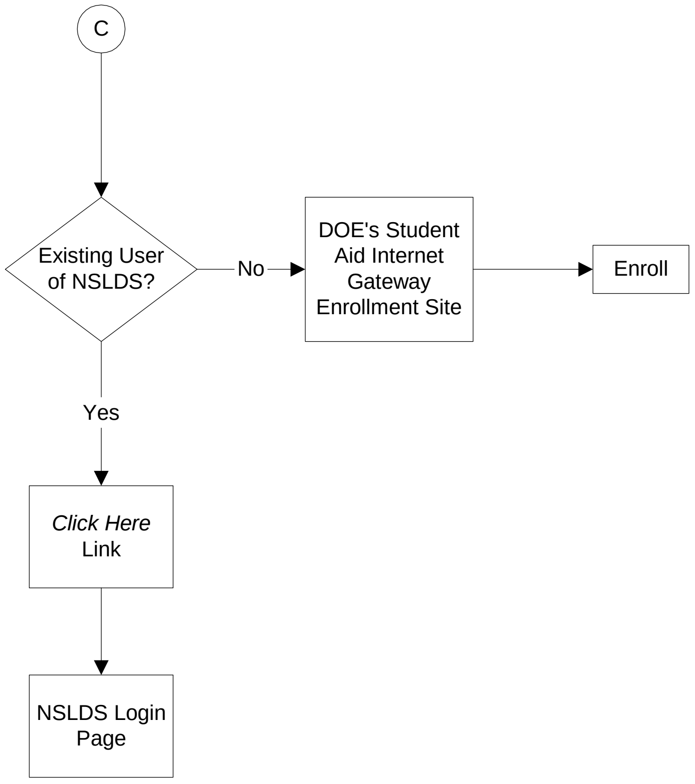

Figure 6

2.3.5 Community Members Search

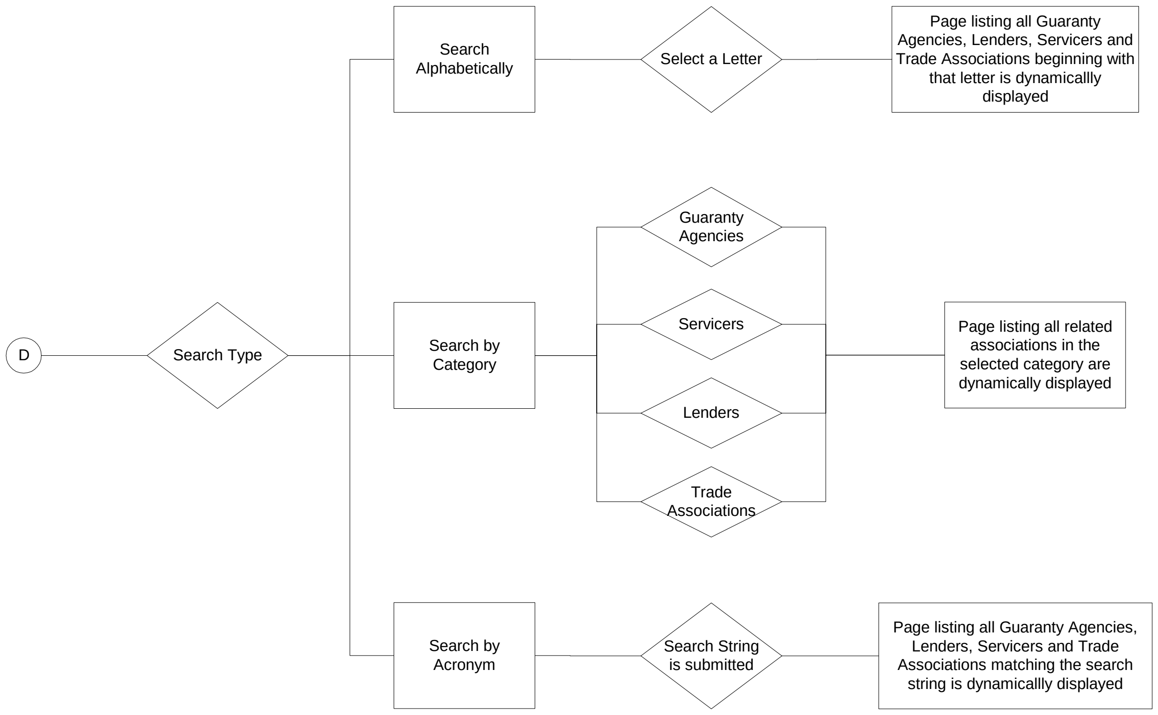
D
Search Type
Search Alphabetically
Select a Letter
Page listing all Guaranty Agencies, Lenders, Servicers and Trade Associations beginning with that letter is dynamicallly displayed
Search by Category
Guaranty Agencies
Servicers
Lenders
Trade Associations
Page listing all related associations in the selected category are dynamically displayed
Search by Acronym
Search String is submitted
Page listing all Guaranty Agencies, Lenders, Servicers and Trade Associations matching the search string is dynamicallly displayed

Figure 7

2.3.6 Finding FP Contacts

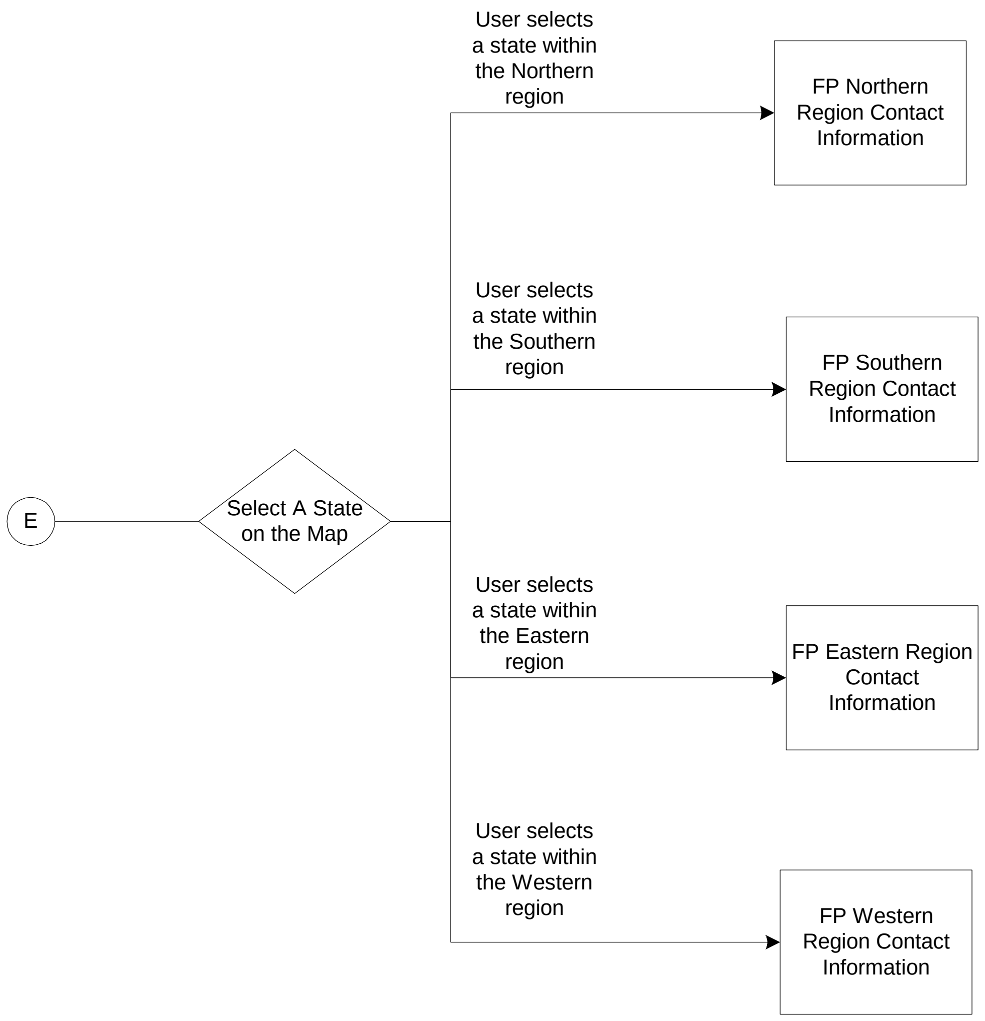
E
Select A State on the Map
User selects a state within the Northern region
FP Northern Region Contact Information
User selects a state within the Southern region
FP Southern Region Contact Information
User selects a state within the Eastern region
FP Eastern Region Contact Information
User selects a state within the Western region
FP Western Region Contact Information

Figure 8

2.4 Audience & Users

The FP portal is to target guaranty agencies, lenders, servicer, trade associations and FSA internal users. For the first release, users will be directed to access FMS (Form 2000, LEAP and SLEAP), Data Mart, and NSLDS through the portal. In addition, users will be able to access Trade Associations, regulatory information content, a FP organization chart, contact information, and FSA University for conference registration via the links through the portal.

2.5 Multi Language

The FSA student portal has the need for multi-lingual capabilities. These languages will include both English and Spanish with English being the default language. The approach to handling multi-language will consist of the following:

Java uses the concept of properties files to store static nave/value pairs. There must be a separate file for each supported language and it must conform to the following naming convention:

- <filename>_<language code>.properties

See the example below:

- **resources_es.properties**

It is also recommended to have two files for the default language. In this case the default language is English, thus there should be a properties file entitled:

- <filename>_<language code>.properties *[default with language code]*

See the example below:

- **resources_en.properties**

and one with no underscore and language code:

- <filename>.properties *[default without language code]*

See the example below:

- **resources.properties**

This last example will always be used by the application if the set locale cannot be linked to a resource properties file.

Struts provides the flexibility to automatically detect the browser's locale. This set locale is used to "point" the application to the correct properties file. Therefore all static text will be stored in this property file and adjusting the locale in either the browser or via a button on the portal will change the locale session variable resulting in a page "refresh" with the new language displayed.

3 Technical Architecture

3.1 Portlets

The FSA Portal site is composed of a horizontal portal, implemented as a frameset, linking to a set of individual vertical portlet action objects. Each portlet in a horizontal portal is generally a separate MVC (Model-View-Controller) application, loosely linked to the overall portal by ancillary requirements, such as security and visual integration (common look and feel). For productivity purposes it is desirable to partition the work so that a team of developers can work on portlets independently. Key to this is to have a framework available that provides a common set of MVC facilities, including:

- A navigation framework, which standardizes the mechanism used to route request processors and the target of those requests (JSP or HTML). In Struts the request processors are termed Actions.

- JSP Tag Libraries, which simplify the development of the portlet JSPs, since they reduce the amount of JSP scripting that needs to be provided. Struts provides a number of Custom Tag libraries. For example, the tag library eases specification of HTML forms by providing a set of message display facilities, buttons, and other forms-based visual controls. Portlet developers may further extend the set of tags to provide enhanced functionality as well as common look and feel between portlets.

- A simple means of binding portlet JSP parameters to Actions by means of some Java Bean having a set of properties, each one corresponding to an HTML form or query parameter. In Struts the parameter-binding object is termed a Form. In addition, Struts provides a standardized validation mechanism for HTML form parameters, very useful in forms intensive portlets.

Using Struts, the portlet developer's role is thus greatly simplified to providing Struts Actions, Forms, JSPs and JSP tags and integrating them by developing some additional Java Beans. All the portlet JSPs will be written using a combination of HTML, Struts tags, and FSA custom tags.

The Integrated Technical Architecture (ITA) team will develop the following 7 portlets for the FSA Portal:

- Login
- Search
- Personalization
- Headlines
- Feedback
- Calendar
- 0Registration

However, Financial Partners, for Release 1 will only be using a subset of these portlets. The ones being used for Release 1 are described in more detail below.

- Search

  The search portlet provides searching capabilities across many sources by providing general and advanced search options using Autonomy. Users can search across the portal pages to find relative information within the portal. Users can search across any or all Dept. of Ed. Sites. Therefore, they have the ability to choose (checkbox) what sites they wish to search across. Sites we'll need to search (ed.gov, FAFSA, Loan Consolidation, Direct Loans, NSLDS, IFAP, *.ed.gov). Users can have the ability to search any of Dept. of Ed's databases. Search results will include a percentage of how close the results are to the search criteria they provided. The search engine will provide results with similar options. This would consist of pages that are contained within that URL. Search results will supply a list of related areas based upon the users search criteria.

Ex: a user searches for space….the browser returns related searches for space pictures, space shuttle, space station, etc. If a user spells a word incorrectly the search engine will signal correct spellings (e.g. did you mean <>?). User can use an advanced search to help find the information they are looking for. This would include a form specifying different criteria.

- Headlines

  The headlines portlet provides messages/headlines from the database to the user. The user sees enterprise-wide news/headlines which is information stored in a database. Users see news based upon what channel (students, FP, schools) they are on in the portal. Thus, if a user is in the students channel the news is geared toward students (not the enterprise). Person(s) can enter headlines via a web based user interface. If personalized messages are used this person can relate it to certain criteria (e.g. state, age, etc). If not personalized then this person simply enters news which is published to the site. Headlines when entered have a start and end date. Therefore the portlet will know when headlines are old and should expire. There will always be at least one headline that will not expire.

- Feedback

  The feedback portlet provides the user with a feedback (or survey) form to solicit feedback on the site from the user. Users when choosing to provide feedback will be taken to a custom questionnaire. This questionnaire can be customized (extended/contracted) by number of questions, question, and possible answers. There are free text fields in this case. This results, upon submission, will be mailed to a specified inbox and stored in the system's DB. A person(s) from the department will be able to add, subtract, modify the current live questionnaire. This will alter the current feedback area and new feedback can be obtained. Results will still be mailed to a specific e-mail address. Upon feedback submission the user will receive a 'Thanks' page.

- Calendar

  The calendar portlet provides the user with calendar information in different views (monthly, weekly, etc). The user can see a month view with links of that days events. Users have links to change months (backwards and forwards). If a user selects a different month the calendar moves to that month and displays events for that day (the day stays the same as the current). If the user clicks the month button they are taken to an exploded view with all events for that month. If a user clicks a day that day is highlighted and entries for that day are shown. Also the user can toggle between different categories (events, deadlines, training, NPRM's). If a user clicks on a certain category the exploded month view is shown with all related activities for that month. Person(s) can add entries to the calendar system as needed via a web based interface. These entries might be categorized by Schools, FP, and Students so each area sees different events.

3.2 Content Management

Content Management will be used to ensure content can be easily managed on the financial partner’s portal. Initially the solution will be a simplified one due to the amount of content and the frequency it needs to be updated. However in future releases the portal will most likely utilize a software product to handle entering web content.

3.3 ITA – Reusable Common Services (RCS)

The Integrated Technical Architecture (ITA) provides the Department of Education's Student Financial Assistance (FSA) program with a more robust core infrastructure for its application and production efforts. In addition to providing FSA's application teams with a core set of products and Subject Matter Expert (SME) support, ITA provides a set of Java 2 Enterprise Edition (J2EE) technical architecture Reusable Common Services (RCS). These common services have been identified to provide significant value to FSA web development efforts.

The RCS services provide FSA with a core set of Java services that may be used across the enterprise to handle core architectural functions such as exception handling, logging, and e-mail. Future releases of the ITA will provide additional RCS services and enhancements.

From its inception, one of ITA's guiding modernization goals has been to create a reusable middleware architecture that could be leveraged by current and future modernization applications. The ITA R2.0 RCS services are the first ITA deployed components of a reusable production-ready middleware architecture.

The following ITA Reusable Common Services will be utilized in the FSA Portal application:

- E-mail Framework
- Exception Handling Framework
- Logging Framework
- Persistence Framework
- Search Framework

The RCS services are built based on an open technology and J2EE architecture. The ITA team leveraged previous Accenture and industry experience to design and develop a robust set of core technical architecture common services that specifically address FSA enterprise application requirements.

The core of the RCS services consists of the exception handling and logging frameworks. The other frameworks use these services to provide standard and consistent exception handling and logging.

3.3.1 E-mail Framework

The e-mail framework provides FSA with a common way to generate e-mail messages from applications. The e-mail framework uses Sun Microsystems' JavaMail API 1.2, which provides a standard interface for Java programs to send e-mails to a Simple Mail Transport Protocol (SMTP) Mail server.

The e-mail framework may be used by FSA application teams to standardize e-mail messaging and replace existing methods of sending e-mail. (The ITA R1.0 applications currently send e-mail through two methods – one uses JavaMail and the other uses an Oracle database, UNIX shells scripts, and a sendmail operating system utility.)

The ITA e-mail framework provides the following features:

- Dynamically set all elements of an e-mail ("To" address, "From" address, subject, etc.)
- Send attachments to e-mail
- Set multiple e-mail addresses within the "To" address, "From" address, and "Reply To" address
- Dynamically set the SMTP Server
- Verify e-mail parameters meet minimum standards for delivery.
- Send a real time e-mail to a SMTP Server
- Process batch e-mails and send to a SMTP Server at a specified time

3.3.2 Exception Handling

An exception is a code or language construct that indicates when an unusual or unexpected error condition occurs in an application. Examples of exceptions are hardware, network, I/O, or memory problems. If an exception is "handled" in code, it can be dealt with gracefully and will not necessarily have to cause program termination. Exception handling provides a mechanism for writing robust, resilient code that is capable of dealing with the unexpected.

The exception handling framework will help standardize and simplify exception handling for FSA's application teams. The standardized exception handling will also help reduce the possibility of uncaught exception scenarios.

The exception handling framework provides the following general services and components:

- Guidelines for identification and responding to exceptions
- Guidelines for throwing and catching exceptions
- Base exception classes
- Default last-resort exception handlers
- Simple interface for integration of logging exceptions
- A generic exception class that must be thrown by all components in the application. It contains a status code that represents the type of exception. This generic exception can be extended for specific errors

An exception factory class that will be used to create exceptions and will automatically assign a unique id. This unique id will be displayed to the user if necessary in order to uniquely identify the associated log and therefore all the associated information.

3.3.3 Logging Framework

The logging framework enables users to track and identify the source of errors. The logging framework uses a routine to determine the originating class and method for the logging call. This provides complete and descriptive logs that enable operations personnel to view the logs and quickly determine where the instance or error occurred and possibly what caused it. Logs are not tied to exceptions only; they are customized to record access and other useful information in troubleshooting and analyzing an application.

Users can define handlers to be global or assign them to specific, named loggers. Loggers can be associated to both the global handler set and to specific handlers. The formatting of the message only happens at the handler. Both loggers and handlers can filter messages based on some function, and by the level of the message.

A set of appropriate handlers will be defined for the given application. This could mean Error or higher goes to a specific handler and Info and higher goes to another. In general, these handlers will be global. Application code should log to a named logger appropriate to its subsystem. Finer control of log levels is set within the handler. However, further filtering will require the creation of a custom filter that is attached to the handler.

The logging framework provides the following features:

- Custom logging
- Filtering messages by level
- Integration with the exception handling framework
- Channel functionality to provide the ability to listen to multiple applications on one server

3.3.4 Persistence Framework

The persistence framework encapsulates the behavior needed to make objects persistent. Specifically, a persistence framework reads, writes, and deletes objects to/from permanent storage. The persistence provides full encapsulation of the persistence mechanism. Application developers can send a save, delete or retrieve message to the persistence framework and the framework will handle the rest of the interaction with the database.

The persistence framework also provides the ability to implement persistence behavior on multiple objects concurrently. The framework supports saving, deleting, or retrieving many objects at once depending upon a specific criterion.

The persistence layer can implement transactional behavior on objects. A transaction is defined as a combination of actions implemented on several objects concurrently. An example is adding an object to a database and deleting another object from another database and being able to rollback the entire transaction if an error occurred.

The persistence layer uses pooling resources available to help maintain efficient use of the database. If a single client has the ability to request every record from a data source, then that client may be able to consume almost all resources of that data source. The persistence framework uses a controlled approach that does not allow runaway use of a resource.

The persistence layer can dynamically run stored procedures on the database or submit SQL directly from the application. The persistence framework includes application supplied data classes that allow the framework to know the schema of the database to which it is connected.

3.3.5 *Search Framework*

The search framework simplifies, standardizes, and improves the use of the Autonomy search engine. This framework complies with J2EE standards instead of using CGI as in the current search engine interface. The framework consists of a search classes that provides a common way to access the Autonomy HTTP API and utilize its features.

The search wrapper implements the following Autonomy features:

- Query search engine
- Natural Language or "Fuzzy" query search engine
- Display search results
- Suggest additional search results

3.4 Software

Function	Product	Version	Environment
Operating System	Sun Solaris	v. 2.6	All
HTTP Server	IBM HTTP Server	v. 1.3.12.2	All
Java Application Server	WebSphere Application Server Advanced Edition	v. 3.5.3	All
Search Engine	Autonomy Knowledge Server	v. 2.1	All
Database	Oracle 8i	v. 8.1.6	All
Java Development Tool	Visual Age for Java Enterprise Edition	v. 3.5.3	Dev Only

Configuration Management and Version Control Tool	Rational ClearCase	v. 4.1	Dev & System Testing
Defect and Change Tracking Tool	Rational ClearQuest	v. 2001A.04.00	System & Performance Testing
Performance Testing Tool	Mercury LoadRunner	v. 7.0	Performance Only

3.5 Environments

Four environments will be used in the life cycle of the FSA Portal Students and Financial Partners Channels.

- Development
- Testing
- Performance Testing
- Production

The following diagrams are logical representations of each of these four environments.

3.5.1 Development

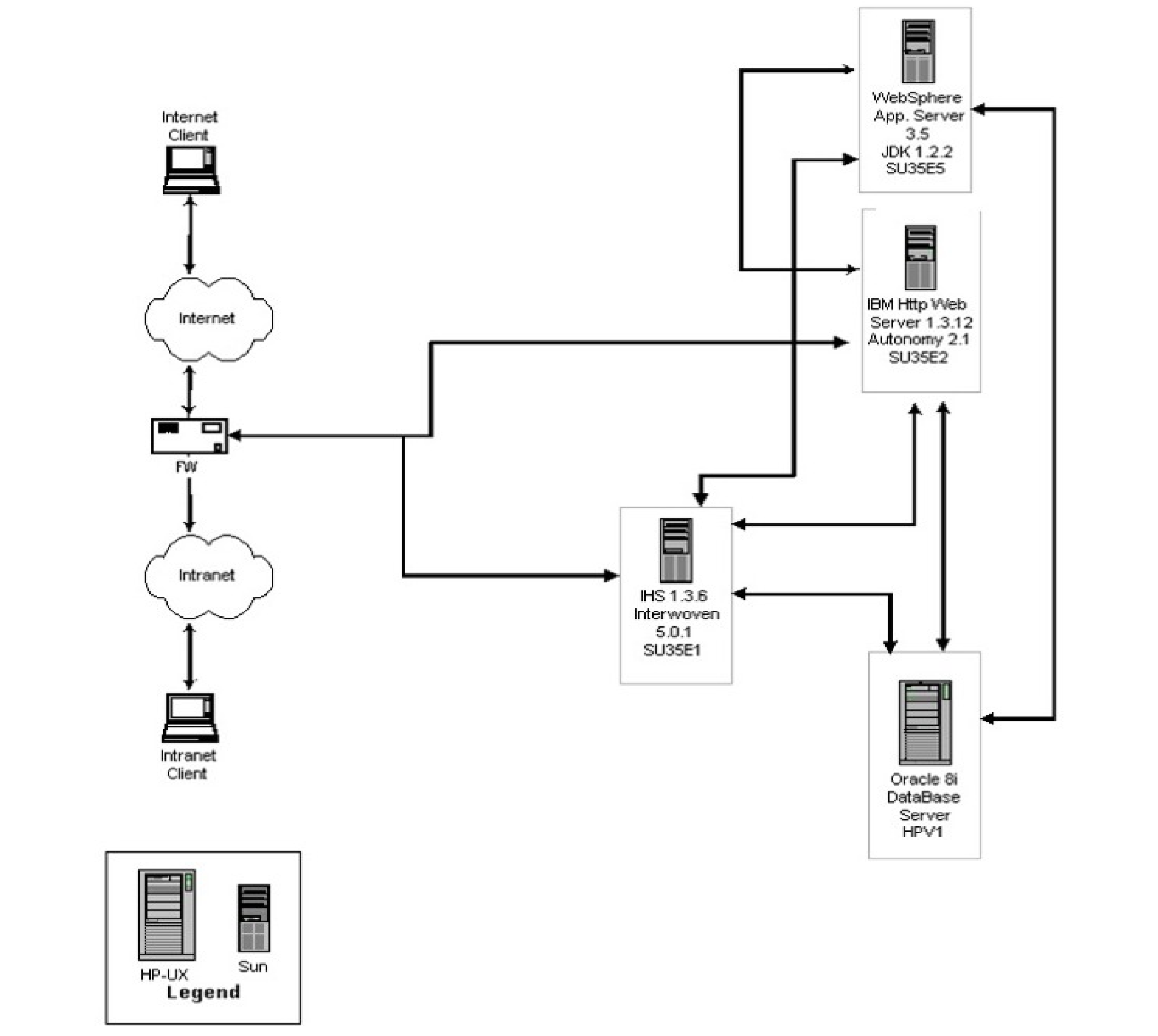
Internet Client
Internet
FW
Intranet
Intranet Client
WebSphere App. Server 3.5 JDK 1.2.2 SU35E5
IBM Http Web Server 1.3.12 Autonomy 2.1 SU35E2
IHS 1.3.6 Interwoven 5.0.1 SU35E1
Oracle 8i DataBase Server HPV1
HP-UX
Sun
Legend

Figure 9

Development Environment (Logical Diagram)

3.5.2 System Testing

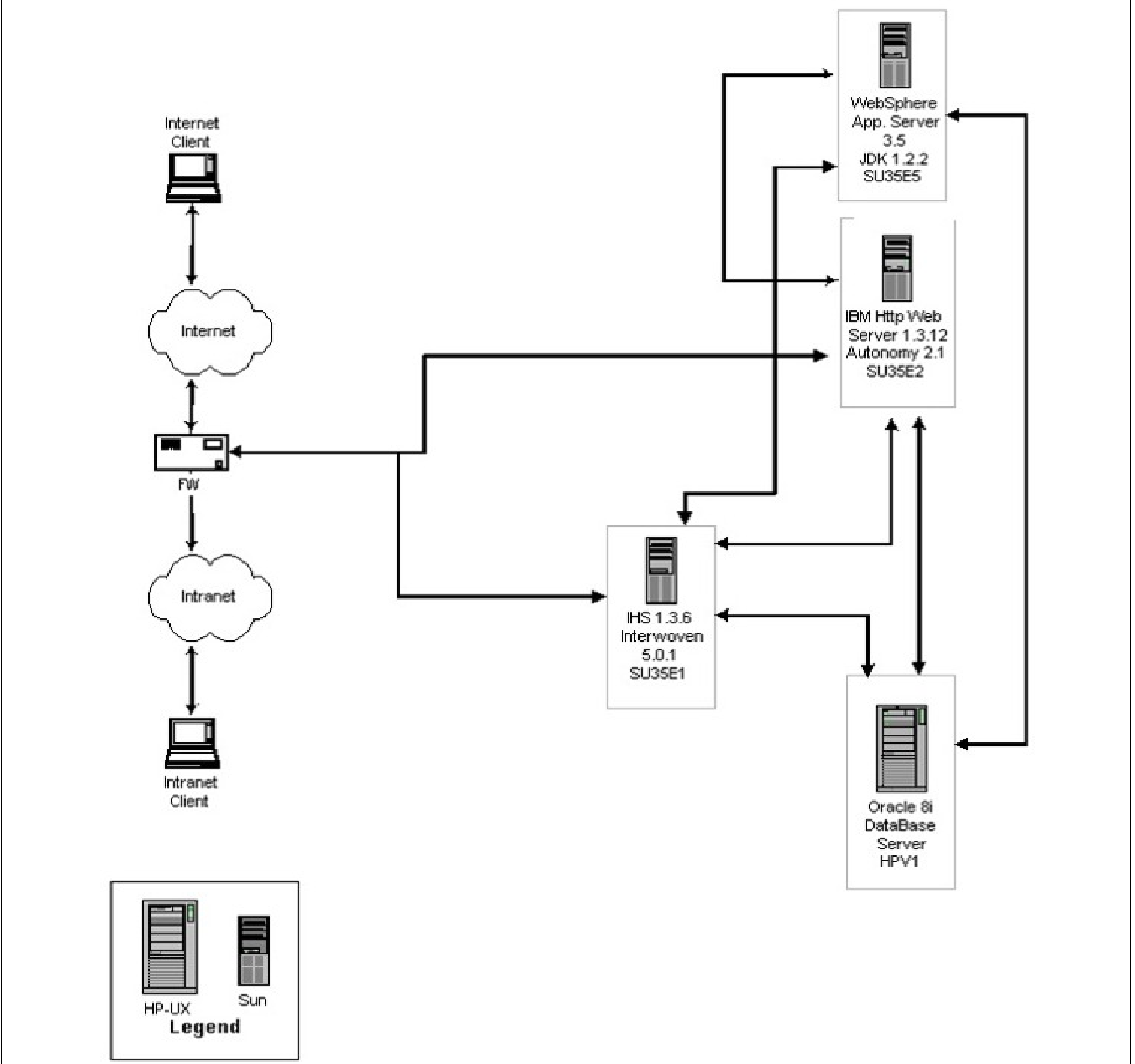
Internet Client
Internet
FW
Intranet
Intranet Client
WebSphere App. Server 3.5 JDK 1.2.2 SU35E5
IBM Http Web Server 1.3.12 Autonomy 2.1 SU35E2
IHS 1.3.6 Interwoven 5.0.1 SU35E1
Oracle 8i DataBase Server HPV1
HP-UX
Sun
Legend

Figure 10

System Testing Environment (Logical Diagram)

3.5.3 Performance Testing

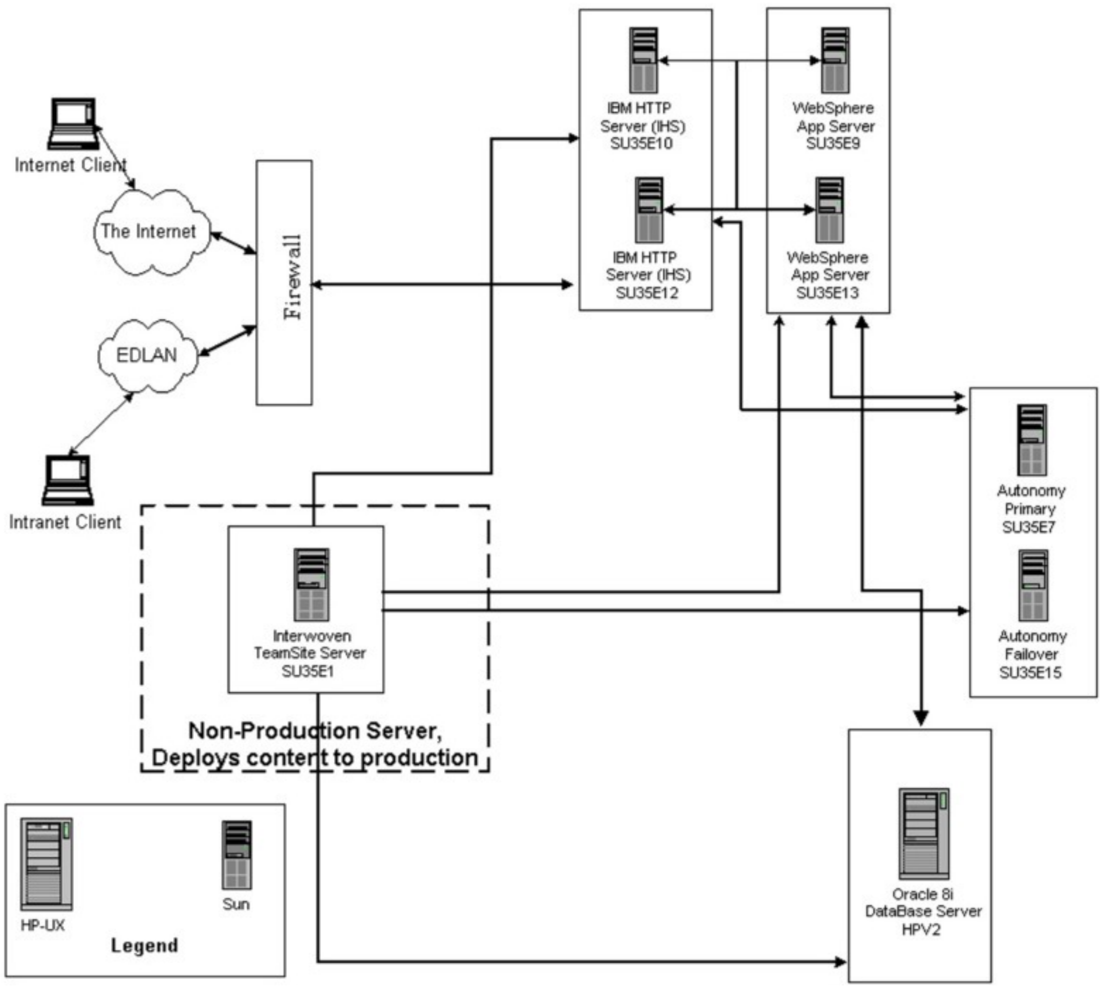
Internet Client
The Internet
EDLAN
Intranet Client
Firewall
IBM HTTP Server (IHS) SU35E10
IBM HTTP Server (IHS) SU35E12
WebSphere App Server SU35E9
WebSphere App Server SU35E13
Autonomy Primary SU35E7
Autonomy Failover SU35E15
Interwoven TeamSite Server SU35E1
Non-Production Server, Deploys content to production
Oracle 8i DataBase Server HPV2
HP-UX
Sun
Legend

Figure 11

Performance Testing Environment (Logical Diagram)

3.5.4 Production

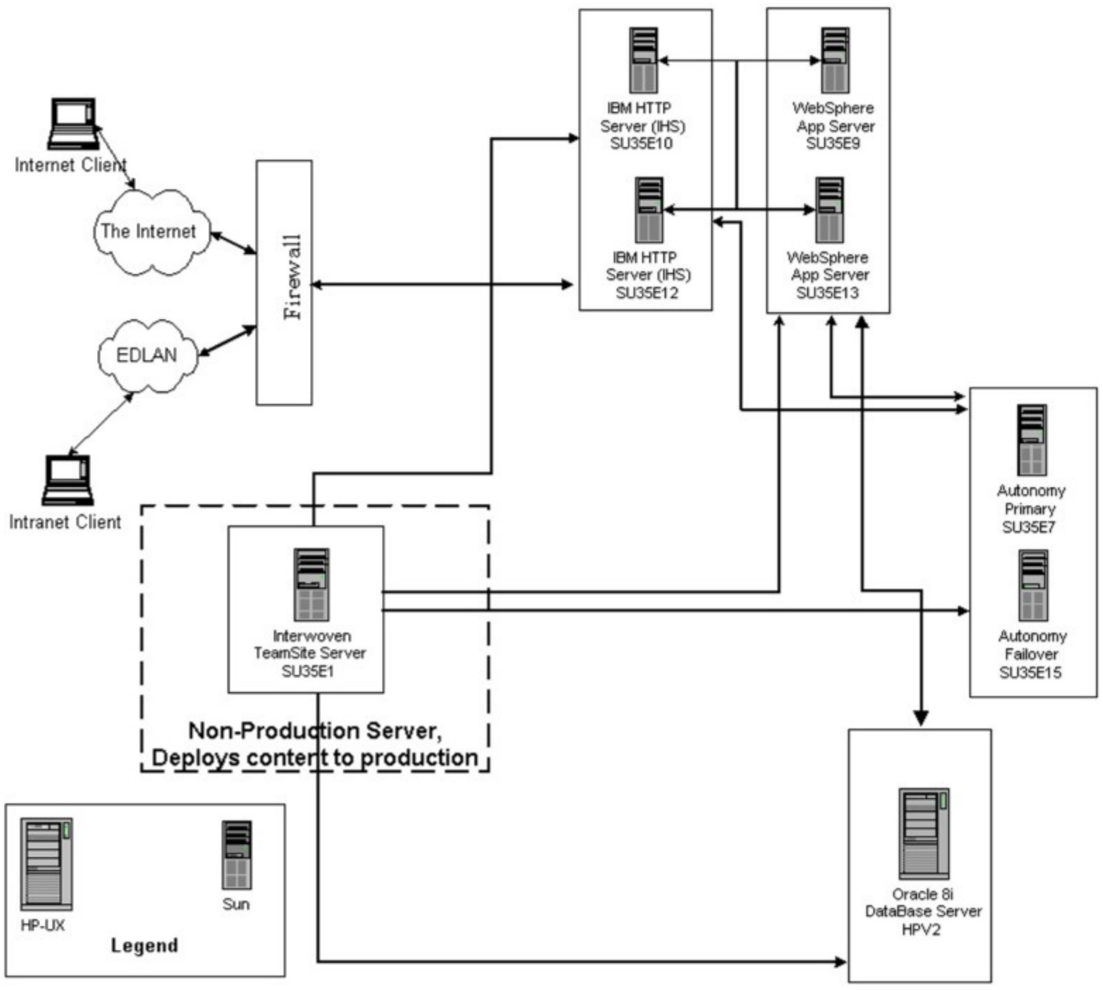
Internet Client
The Internet
EDLAN
Intranet Client
Firewall
IBM HTTP Server (IHS) SU35E10
IBM HTTP Server (IHS) SU35E12
WebSphere App Server SU35E9
WebSphere App Server SU35E13
Autonomy Primary SU35E7
Autonomy Failover SU35E15
Interwoven TeamSite Server SU35E1
Non-Production Server, Deploys content to production
Oracle 8i DataBase Server HPV2
HP-UX
Sun
Legend

Figure 12

Production Environment (Logical Diagram)

3 Data Architecture

4.1 Environment

The infrastructure for the data will consist of an Oracle database. For development and production this instance will reside at the VDC.

4.2 Instance

The instance for the FP portal will contain data for the following areas:

- Headlines

Headlines will contain information relating to a particular headline. This will include the headline text, title, start date, end date, category, audience, etc. Headlines information will be fed via a web based interface thus making the updating process easier and more intuitive.

- Calendar

Calendar data will consist of data for training, deadlines, and events. Each headline will have data related to the URL it related to and the title of the headline. It will also have data pertaining to the start date, end date, cost, audience, and category.

- Feedback

The data related to the feedback portlet consists of the question and its related answer. All other data such as the questions and answers on the site will be stored in a file not a relational database.

- Community Search

The community search feature will hold data for GA's, lenders, servicers, and trade associations. Therefore, the database will hold the organization name, its URL, the category it belongs in, and its acronym.